New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics Volume II: Applications

Editors

Krassimir T. Atanassov
Władysław Homenda
Olgierd Hryniewicz
Janusz Kacprzyk
Maciej Krawczak
Zbigniew Nahorski
Eulalia Szmidt
Sławomir Zadrożny

SRI PAS  **IBS PAN**

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics Volume II: Applications

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics
Volume II: Applications

Systems Research Institute
Polish Academy of Sciences

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics
Volume II: Applications

Editors

Krassimir T. Atanassov
Władysław Homenda
Olgierd Hryniewicz
Janusz Kacprzyk
Maciej Krawczak
Zbigniew Nahorski
Eulalia Szmidt
Sławomir Zadrożny

IBS PAN  **SRI PAS**

Systems Research Institute
Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland
www.ibspan.waw.pl

ISBN 83-894-7541-3

Dedicated to Professor Beloslav Riečan on his 75th anniversary

Model checker of object-oriented programs based on generalized nets

Magdalina Todorova

Faculty of Mathematics and Informatics, Sofia University,
5 "J. Bouchier" Str., Sofia, Bulgaria
todorova_magda@hotmail.com

Abstract

The article presents the structure of model checker for verification of object-oriented programs and an algorithm to implement the model checker. The algorithm of model checker, as well as its input components (function, which is subject to verification and its specification) are defined by means of generalized nets. To simplify the description the algorithm of the model checker is presented in the case when the object-oriented program consists of one class and a main function that uses the class.

Keywords: object-oriented programming (OOP), verification of object-oriented programs, generalized net (GN).

1 Introduction

Subject of consideration in this paper is an implementation of an approach for verification of object-oriented programs. Verification is on specification, defining all the possible correct links between the member functions of the class. The approach integrates the concept of design by contract with approaches to verification of a type proof of theorems, and consistency checking. In it the GNs are used as a means of setting of the specification on which verifies the functions of object-oriented programs. By means of GNs are presented also the functions, subject to verification, and the algorithm, which implemented the check for consistency.

The choice of GNs as a tool for modeling was made for the following reasons:

- GNs are convenient to define the specifications on which the verification is done. The reason is that the specifications we use in the verification ap-

New Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Volume II: Applications *(K.T. Atanassov, W. Homenda, O. Hryniewicz, J.Kacprzyk, M. Krawczak, Z. Nahorski, E Szmidt, S. Zadrożny, Eds.), IBS PAN - SRI PAS, Warsaw, 2012.*

proach are network structures, which set all possible states in which an object of the class may fall, and all sequence of the correct calls to member functions of the class. The places of the GN present states and transitions – possible to run in a given state member functions of the class.

- With the already existing methodology for building GN [1, 2] and with the help of the patterns for representing operators of the OOP language, a GN can be build which corresponds to the function of the given object-oriented program.
- GNs are convenient means for modeling of the process of defining classes [3].
- GNs give means to model parallel processes. This property allows effective execution of GNs in parallel.
- There exists a programming environment GN Lite, with which are executed GNs, which model real-world processes.

2 Structure of the model checker

In [4], an approach for building correct object-oriented programs, realized in C++ programming language, is described. The essence of the approach consists of separating the verification of the class from the verification of the function that applies this class. The verification of the class is performed according to the methodology described by Meyer in [5] and [6], constructing the verifying statements by applying the Floyd-Hoare style logic [7]. The proofs of the verifying statements are executed by manually applying the technique of the transformation predicates [8] or by applying some of the systems for automatic proof of theorems HOL, Coq and others. For the verification of the function which uses the class the following actions are performed:

- A generalized net model of the class is built which defines the connections between the methods in the class in the form of correct sequences of requirements. We will call this net model a formal net project of the class. This model defines the specification by which the verification of the function of the object-oriented program is performed.
- The function of the object-oriented program, which will be verified according to the previously described specification, is represented via a generalized net.
- GNs of the function and the specifications defined from the class are executed in parallel in order to determine if the model of the function corresponds to (complies with) the given specification.

Figure 1 shows the structure of the model checker for object-oriented programs based of generalized nets.

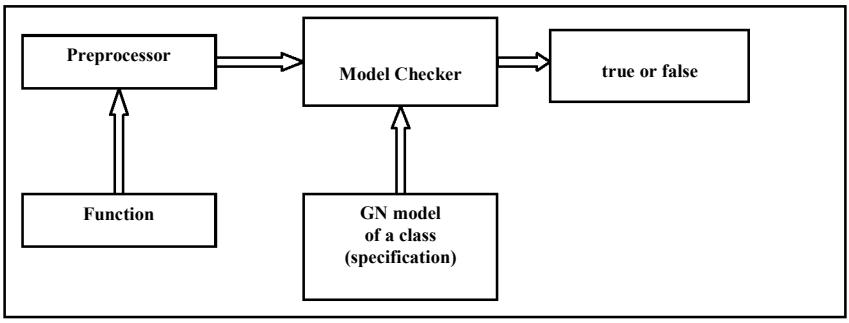

Figure 1: Structure of the Model Checker

The component Function gives the function the object-oriented program, which is to be verified. The Preprocessor extracts the generalized net that corresponds to the function, which will be verified. The model checking engine (Model Checker) takes the generalized net, acquired as a result of the work of the preprocessor and the formal net project of the class, and determines whether the function satisfies the formal specification (i.e. if the two nets are consistent). In the case of a negative result of the check, the model checker reports the reason for the error and shows the place where it has occurred. The creation of the formal net project of the class requires an in-depth knowledge about the class, about the possible connections between its member functions as well as knowledge about the techniques of generalized nets construction. It is realized by the programmer of the class. The algorithm by which the model checker works is described in the following section of the paper.

3 Algorithm for checking of consistency

We will describe the algorithm in the case when the object-oriented program consists of a class *C* and a function *M* that uses it. We will begin with the case when the function that uses the class defines and uses only one object of *C*.

3.1 Variant 1

Let GN_C be the formal net project of the class *C*, and GN_M – the generalized net model of the function *M*. Because we have defined only one object in *M*, each of the nets GN_M and GN_C will have just one token, which will correspond to the object. The net GN_M may contain more tokens, but since they are not connected

with the object of the class C, they are not subject of discussion in this paper. If q is the name of the object in M, then the characteristics of the tokens of GN_M and GN_C will be tuples having the form (q, *name_of_place_in_*GN_M) and (q, *name_of_place_in_*GN_C), respectively.

The generalized net GN_1, described in Figure 2 defines the algorithm for checking whether GN_M corresponds to GN_C. It consists of two transitions. The transition X_1 performs the initialization actions, and X_2 – the check for consistency. The net GN_1 will also have only one token. We define its characteristic as a sequence in the form:

"(*name_of_the_token*, *name_of_place_in_*GN_1, *name_of_place_in_*GN_M, *name_of_place_in_*GN_C)",

where the *name_of_the_token* in the case is q, *name_of_place_in_*GN_1 is the name of the place of the token q in GN_1, and *name_of_place_in_*GN_M and *name_of_place_in_*GN_C are the names of the places of the tokens with the same name in GN_M and GN_C, respectively.

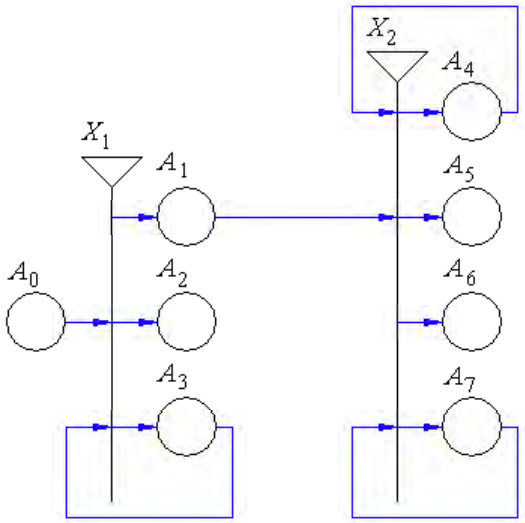

Figure 2: Algorithm for checking if GN_M corresponds to GN_C

There follow the definitions of the transitions in GN_1.

$$X_1 = <\{A_0, A_3\}, \{A_1, A_2, A_3\}, r_1>$$

$$r_1 = \begin{array}{c|ccc} & A_1 & A_2 & A_3 \\ \hline A_0 & P_1 & P_2 & P_3 \\ A_3 & P_1 & P_2 & P_3 \end{array},$$

where:

- P_1 = The current transition to be performed in GN_M and GN_C contains a condition from the type "the member function is a constructor of *C*" and can be executed in parallel.
- $P_2 = \neg P_1$ & $\neg P_3$
- P_3 = The current transition to be performed in GN_M does not contain a condition, connected with a member function of the class *C*.

$$X_2 = <\{A_1, A_4, A_7\}, \{A_4, A_5, A_6, A_7\}, r_2>$$

$$r2 = \begin{array}{c|cccc} & A_4 & A_5 & A_6 & A_7 \\ \hline A_1 & P_4 & P_5 & P_6 & P_7 \\ A_4 & P_4 & P_5 & P_6 & P_7 \\ A_7 & P_4 & P_5 & P_6 & P_7 \end{array},$$

where:

- P_4 = The tokens of GN_M and GN_C are into input places for transitions, which can be executed simultaneously because of existing conditions of the transitions which hold simultaneously.
- P_5 = The execution of the nets GN_M and GN_C is over and they do not contain any active tokens.
- $P_6 = \neg P_4$ & $\neg P_5$ & $\neg P_7$
- $P_7 = P_3$.

The execution of GN_1 begins with the activation of a token at place A_0, input for the transition X_1, which has no characteristics. In the same time, the execution of the nets GN_M and GN_C begins. In their input places tokens without initial characteristics are also activated.

Execution of transition X_1

In case that the predicate P_1 holds, the transition X_1 of GN_1 and the current transitions on GN_M and GN_C are executed. As a result, the tokens of GN_M and GN_C are transfered to the corresponding output places for the executed transitions. Let us denote the names of the new places as M_i and S_j. The tokens of GN_M and GN_C obtain the characteristics (q, M_i) and (q, S_j), respectively. The token of GN_1 is transfered to place A_1 and gets as a characteristic the sequence (q, A_1, M_i, S_j), containing the name of the object and the names of the places A_1, M_i and S_j, where the tokens in the nets GN_1, GN_M and GN_C are, respectively.

The execution of the transition X_1 ends, after which there the execution of the transition X_2 on GN_1 follows.

In the case when the predicate P_3 holds, the transition X_1 on GN_1 and the current transition on GN_M are executed. The transition of GN_C, which has the token as its input place is not executed. As a result, the token of GN_M is transfered to the corresponding output place and does not have a characteristic. The token of GN_C is not transfered. The token of GN_1 is transfered to place A_3 and does not have a characteristic. Because the place A_3 is input for the transition X_1, the next step in the execution of GN_1 is the repeating execution of X_1.

In the case when predicate P_2 holds, i.e. the predicates P_1 and P_3 do not hold, the generalized net model GN_M does not correspond to GN_C and the execution of GN_1 is over. The conditions of the transition X_1 show the reason for the inconsistency.

Execution of transition X_2

In the case that the predicate P_4 holds, the transition X_2 and the transitions with input places containing the tokens of GN_M and GN_C are executed. As a result, the tokens of GN_M and GN_C are transfered and we denote their new places as M_n and S_m, respectively. Their characteristics have the form (q, M_n) and (q, S_m). The token of GN_1 is transfered to place A_4 and gets as a characteristic the sequence (q, A_4, M_n, S_m), containing the name of the object and the names of the places A_4, M_n and S_m, into which the tokens in the nets GN_1, GN_M and GN_C are, respectively. Since place A_4 is input place for transition X_2, the execution of the net GN_1 continues with the next execution of the transition X_2.

In the case when the predicate P_7 holds, the transition X_2 on GN_1 and the transition of GN_M, which obtains as an input the place of the token, are executed. No transition of GN_C is being executed. As a result, the token of GN_M is transfered into output place for the perfomed transition, which we denote by M_k. The token of GN_C remains in its current place and let us denote it by S_l. The token of GN_1 is transfered to place A_7 and gets the characteristic (q, A_7, M_k, S_l). The execution of GN_1 continues with the execution of the transition X_2.

In the case when the predicate P_5 holds, the token of GN_1 is transfered to place A_5. In this case, the generalized net of the function M corresponds to the formal net project of the class C.

When the predicate P_6 holds, i.e. the conditions P_4, P_5 and P_7 do not hold, GN_M does not correspond to GN_C. The token of GN_1 transfered to place A_6 and it changes only the second element of its characteristic. The conditions of the transition give the reasons for the inconsistency, and the characteristic of the token into place A_6 shows the places into GN_C and GN_M, where that inconsistency has occurred.

3.2 Variant 2

In this variant in the function *M* more than one instance of the object of the class *C* have been defined. The generalized net GN_2, described in Figure 3, defines the algorithm to check whether GN_M corresponds to GN_C. In this case, the nets GN_2, GN_M and GN_C will contain as many tokens, related to objects, as is the number of the defined objects in function *M*. The net GN_M may contain more tokens, but since they are not connected with objects of the class *C*, they are not subject of discussion in this paper. Let us denote the sets of the tokens of GN_2, GN_M and GN_C by *K*, K_M and K_C, respectively. The characteristics of the tokens of K_M and K_C are tuples with the following form:

"*(name_of_token, name_of_place_in_*GN_M*)*"
and "*(name_of_token, name_of_place_in_*GN_C*)*"

respectively, and the characteristic of a token of *K* is a sequence in the form:

"*(name_of_token, name_of_place_in_*GN_2*, name_of_place_in_*GN_M*, name_of_place_in_* GN_C*)*",

which gives for each token its name, the name of the place in GN_2 and the names of the places of its corresponding tokens (tokens with the same names) into GN_M and GN_C. The priority of the tokens is determined by the order in which the objects are defined and used in the function *M*. Each transition in the net GN_M, which is connected with a call to the member function of the class *C*, is connected also with the name of the object, for which the call to the member function is performed. The following execution of such a transition in GN_M determines the name of the active tokens in the three nets.

The net GN_2 is composed of two transitions. The transition X_1 performs the initialization actions, and X_2 checks for consistency.

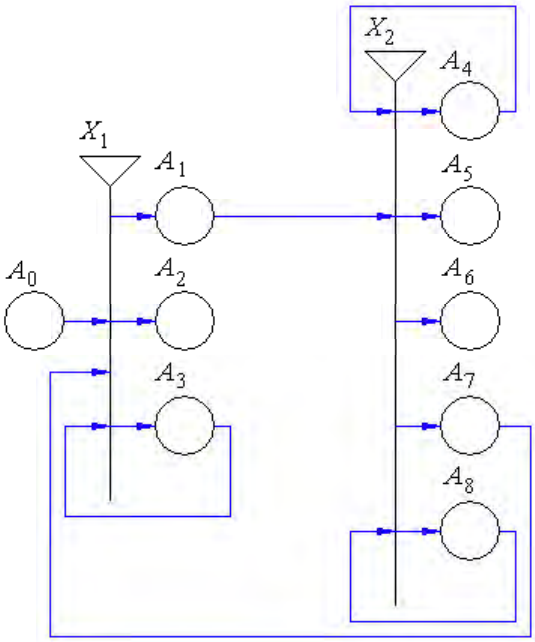

Figure 3: Algorithm to check if GN_M corresponds to GN_C

The definitions of the transitions of GN_2 are as follows.

$$X_1 = <\{A_0, A_3, A_7\}, \{A_1, A_2, A_3\}, r_1>$$

$$r_1 = \begin{array}{c|ccc} & A_1 & A_2 & A_3 \\ \hline A_0 & P_1 & P_2 & P_3 \\ A_3 & P_1 & P_2 & P_3 \\ A_7 & P_1 & P_2 & false \end{array},$$

where:

- P_1 = The transitions with input places, in which the activated tokens of GN_C and GN_M are, contain a condition from the type "the member function is a constructor of C" and can be executed in parallel.
- $P_2 = \neg P_1 \,\&\, \neg P_3$
- P_3 = The activated token of GN_M is into place which is an input for a transition, which does not contain a condition, connected with a member function of the class C.

$$X_2 = <\{A_1, A_4, A_8\}, \{A_4, A_5, A_6, A_7, A_8\}, r_2>$$

$$r2 = \begin{array}{c|ccccc} & A_4 & A_5 & A_6 & A_7 & A_8 \\ \hline A_1 & P_4 & P_5 & P_6 & P_7 & P_8 \\ A_4 & P_4 & P_5 & P_6 & P_7 & P_8 \\ A_8 & P_4 & P_5 & P_6 & P_7 & P_8 \end{array},$$

where:

- P_4 = The activated tokens of GN_M and GN_C with the same name are into places input for transitions, not executing constructors of the class C and which can be executed simultaneously because of the existence of conditions of the transitions, which hold simultaneously.
- P_5 = The execution of the nets GN_M and GN_C is over and there are no active tokens inside of them.
- $P_6 = \neg P_4 \,\&\, \neg P_5 \,\&\, \neg P_7 \,\&\, \neg P_8$
- P_7 = The active token of GN_M is in a place, which is input for transition, executing a constructor of the class C.
- $P_8 = P_3$.

The execution of GN_2 starts with an activation of a token at place A_0, which is input for the transition X_1, having no initial characteristic. In parallel with this, the execution of the nets GN_M and GN_C starts. In their input places tokens without initial characteristics are also activated.

Execution of transition X_1

In the case that predicate P_1 holds, the transition X_1 and the transitions with input places, where there are activated tokens of GN_M and GN_C, are executed. As a result, the active tokens of GN_M and GN_C are transfered to the corresponding output for the executing transitions places M_i and S_j and get characteristics (q, M_i) and (q, S_j), respectively, where q is the name of the object which is defined after the call of the constructor. The token of GN_2 is transfered to place A_1 and gets as characteristic the sequence (q, A_1, M_i, S_j), containing the name of the object and the names of the places A_1, M_i and S_j, into which stay the activated tokens in the nets GN_2, GN_M and GN_C, respectively. The execution of transition X_1 ends, after which the execution of transition of X_2 of GN_2 takes place.

The case when predicate P_3 holds is similar to this for the GN_1 net.

In the case that the predicate P_2 holds, i.e. the conditions P_1 and P_3 are not fulfilled, the generalized net model GN_M does not correspond to GN_C and the execution of GN_2 is over. The conditions of the transition X_1 provide the reason for the mismatch, and the characteristic of the token (if existing) gives the places into GN_M and GN_C, where it is occurred.

Execution of transition X_2

In the case that the predicate P_4 holds, the transition X_2 as well as the transitions with the input places, in which stay the activated tokens of GN_M and GN_C, are executed. Let the tokens of GN_M and GN_C are transfered to the places M_n and S_m, respectively, and the name of the active token is q. Then their characteristics are (q, M_n) and (q, S_m). The token of GN_2 is transfered to place A_4 and gets the characteristic (q, A_4, M_n, S_m). Since place A_4 is input of transition X_2, the execution of GN_2 continues with the next execution of transition X_2.

In the case that the predicate P_8 holds, the transition of X_2 of GN_2 and the transition of GN_M, which has as input the place into which is the active token, are executed. The transition of GN_C is not executed. As a result, the active corresponding token of M is forwarded to the corresponding exit place (let us denote it by M_k). The corresponding token of GN_C stays in its current place (let us denote it by S_l). The GN_2 token transferes to place A_8 and gets the characteristic (q, A_8, M_k, S_l), where q is the name of the active token. The execution of GN_2 continues with the execution of the transition X_2.

In the case when the predicate P_5 holds, the token of GN_2 is transfered to place A_5 and it loses its characteristic. In this case, the generalized net of the function corresponds to the formal net project of the class.

In the case where the predicate P_7 holds, the current active tokens in the three nets are deactivated. Three new tokens without characteristics that correspond to the object, which would be created after the call to the constructor, are being activated. For this purpose the transition X_2 is executed in the beginning. As a result, in A_7 a new token without characteristic is activated. A token without characteristic is activated in the current input place of GN_M, in which the last activated token used to be, and one more token without characteristic is activated – in the input place of the net GN_C. The next action is the execution of the transition X_1 of GN_2 and the transitions in GN_M and GN_C, corresponding to the constructor.

When condition P_6 holds, i.e. conditions P_4, P_5, P_7 and P_8 of the transition X_2 do not hold, GN_M does not correspond to GN_C. The token of GN_2 transferes to place A_6 and changes only the second element of its characteristic. The conditions of the transition give the reasons for the discrepancy and the characteristic of the token in place A_6 shows the places in GN_C and GN_M, where the inconsistency has occurred.

4 Conclusions

This article briefly presents the structure of realization of a model checker, as well as an algorithm for verification of a function of an object-oriented program

according to the specification, given by the class defined in the program. In its essence, this is the parallel execution of two generalized nets – the functional one and the specification one. The execution continues until an error occurs showing the inconsistency (i.e. the function does not comply with the specifications), or until the smooth completion of the parallel execution of the nets, corresponding to a correctness of the function according to the specification. Although the description is for the simplest object-oriented programs, research done show that generalized nets are a powerful enough apparatus and their usage motivates the continuation of the research in more general cases. A motivation for research in this direction also gives the improvement of software for implementation of GN models, as well as its broadening with new opportunities [9], [10], [11] and [12].

Acknowledgments

The paper is supported by Grant 182/2011 from Sofia University Research Fund.

References

[1] Atanassov, K., On Generalized Nets Theory, Prof. Marin Drinov Academic Publishing House, Sofia, 2007.

[2] Atanassov, K., Generalized Nets, World Scientific, Singapore, 1991.

[3] Sotirova, E., P. Tcheshmedjiev, D. Orozova, Modelling the Process of Defining Java Class Using Generalized Net, Proceedings of the Jangjeon Mathematical Society, Vol. 9, No.2, December, 2006, 201-215.

[4] Todorova, M., Construction of Correct Object-Oriented Programs via Building there Generalized Nets Models, Annual of "Informatics", Section Union of Scientists in Bulgaria, Vol. 4, 2011 (in print).

[5] Meyer, B., Object-Oriented Software Construction, 2nd edition, ISE Inc. Santa Barbara, California, 1997.

[6] Meyer, B., Applying "Design by Contract", Computer Vol. 25, 1992, No 10, 40-51.

[7] Hoare, C.A.R., Proof of Correctness of Data Representations, Acta Informatica, Vol. 1, N 4, 1972

[8] Gries, D., The Science of Programming, Springer-Verlag, Berlin and New York, 1981.

[9] Dimitrov, D. G. A Graphical Environment for Modeling and Simulation with Generalized Nets, Annual of "Informatics", Section Union of Scientists in Bulgaria, Vol. 3, 2010, 51-66.

[10] Dimitrov, D. G., Software Products Implementing Generalized Nets, Annual of "Informatics" Section of the Union of Scientists in Bulgaria, Vol. 3, 2010, 37-50.

[11] Dimitrov, D. G., Optimized Algorithm for Token Transfer in Generalized Nets, Proc. of 9th IWIFSGN 2010, 8 October 2010, Warsaw, Poland.

[12] Atanassov, K., D. Dimitrov and V. Atanassova, Algorithms for Tokens Transfer in the Different Types of Intuitionistic Fuzzy Generalized Nets. Cybernetics and Information Technologies, Vol. 10, 2010, No 4, 22-35.

The papers presented in this Volume 2 constitute a collection of contributions, both of a foundational and applied type, by both well-known experts and young researchers in various fields of broadly perceived intelligent systems.
It may be viewed as a result of fruitful discussions held during the Tenth International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGN-2011) organized in Warsaw on September 30, 2011 by the Systems Research Institute, Polish Academy of Sciences, in Warsaw, Poland, Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences in Sofia, Bulgaria, and WIT - Warsaw School of Information Technology in Warsaw, Poland, and co-organized by: the Matej Bel University, Banska Bystrica, Slovakia, Universidad Publica de Navarra, Pamplona, Spain, Universidade de Tras-Os-Montes e Alto Douro, Vila Real, Portugal, and the University of Westminster, Harrow, UK:

Http://www.ibspan.waw.pl/ifs2011

The consecutive International Workshops on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGNs) have been meant to provide a forum for the presentation of new results and for scientific discussion on new developments in foundations and applications of intuitionistic fuzzy sets and generalized nets pioneered by Professor Krassimir T. Atanassov. Other topics related to broadly perceived representation and processing of uncertain and imprecise information and intelligent systems have also been included. The Tenth International Workshop on Intuitionistic Fuzzy Sets and Generalized Nets (IWIFSGN-2011) is a continuation of this undertaking, and provides many new ideas and results in the areas concerned.

We hope that a collection of main contributions presented at the Workshop, completed with many papers by leading experts who have not been able to participate, will provide a source of much needed information on recent trends in the topics considered.

ISBN-13 9788389475411